Constraints, 8, 365–388, 2003

CTP: A New Constraint-Based Formalism for Conditional, Temporal Planning

IOANNIS TSAMARDINOS ioannis.tsamardinos@vanderbilt.edu
Department of Biomedical Informatics, Vanderbilt University

THIERRY VIDAL thierry@enit.fr
Production Engineering Laboratory (LGP) - ENIT, France

MARTHA E. POLLACK pollackm@eecs.umich.edu
Computer Science and Engineering, University of Michigan

Abstract. Temporal constraints pose a challenge for conditional planning, because it is necessary for a conditional planner to determine whether a candidate plan will satisfy the specified temporal constraints. This can be difficult, because temporal assignments that satisfy the constraints associated with one conditional branch may fail to satisfy the constraints along a different branch. In this paper we address this challenge by developing the Conditional Temporal Problem (CTP) formalism, an extension of standard temporal constraint-satisfaction processing models used in non-conditional temporal planning. Specifically, we augment temporal CSP frameworks by (1) adding observation nodes, and (2) attaching labels to all nodes to indicate the situation(s) in which each will be executed. Our extended framework allows for the construction of conditional plans that are guaranteed to satisfy complex temporal constraints. Importantly, this can be achieved even while allowing for decisions about the precise timing of actions to be postponed until execution time, thereby adding flexibility and making it possible to dynamically adapt the plan in response to the observations made during execution. We also show that, even for plans without explicit quantitative temporal constraints, our approach fixes a problem in the earlier approaches to conditional planning, which resulted in their being incomplete.

Keywords: temporal reasoning, constraint-based planning, conditional planning

1. Introduction

Classical planning [20] assumes that a plan can be generated off-line, prior to its execution, and that execution consists in the straightforward activation of the steps in the plan, in an order that is consistent with the plan's temporal constraints. This assumption is satisfied in domains in which the planning agent is omniscient, and thus knows at plan time everything required about the possible evolution of the world during execution. In many real-world domains, this assumption is violated.

One way to handle this difficulty is to build the plan on-line, making all decisions in a reactive fashion. However, the reactive approach has a number of shortcomings; in particular, when there are real-time requirements to be satisfied, a reactive approach typically cannot guarantee that they will be met (nor can it determine that they are unsatisfiable). In addition, a reactive approach will fail to take preparatory steps that might be required for the continuation of the plan in unexpected circumstances. Consequently, an alternative approach has been to develop conditional planning capabilities [14, 15, 17]. In the conditional planning approach, plans are generated prior to execution, but they include both observation actions and conditional branches,

which may or may not be executed, depending on the execution-time result of certain observations.

In addition to omniscience, classical planning also assumes instantaneous, atomic actions. However, modern planning systems [8, 11] indicate the growing need to represent and reason with metric temporal information and constraints. Temporal constraints pose a challenge to conditional planning. In particular, it is necessary for a temporal conditional planner to determine whether a candidate plan can possibly satisfy the specified temporal constraints. This can be difficult, because temporal assignments that satisfy the constraints associated with one conditional branch may fail to satisfy the constraints along a different branch.

In this paper we address this challenge by developing the **Conditional Temporal Problem** (CTP) formalism, an extension of standard temporal constraint-satisfaction processing models used in non-conditional temporal planning. Specifically, we augment the previous models by (1) adding observation nodes, which correspond to the time-points at which observation actions end, and (2) attaching labels to all other nodes in the network. A node's label indicates the situation(s) in which the event it denotes (the start or the end of a plan's action) will be executed.

As we will show, our extended framework allows for the off-line construction of conditional plans that are guaranteed to satisfy complex temporal constraints. Importantly, this can be achieved even while allowing for the decisions about the precise timing of actions to be postponed until execution time, in a least-commitment manner, thereby adding flexibility and making it possible to adapt the plan dynamically, during execution, in response to the observations made. We also show that, even for plans without explicit quantitative temporal constraints, our approach fixes a problem in the earlier approaches to conditional planning, which resulted in their being incomplete.

In Section 2 we review the temporal constraint models that inspired our conditional model, which itself is depicted in Section 3. Three levels of consistency in conditional temporal problems are then defined in Section 4. Each is developed, in turn, in the three subsequent sections, which provide algorithms, discuss theoretical complexity issues, and describe the usefulness of each form of consistency to planning. In particular, in Section 8, we explain how the use of our approach in conditional planning corrects a problem in the earlier approaches. We conclude this article with a discussion of related and future work.

2. Background

A variety of planning systems [8, 11], often referred to as temporal planners, use an underlying constraint model to query and check the temporal aspects of a plan. One of the most popular models is the *Temporal Constraint Satisfaction Problem* (TCSP) and its graph-based counterpart, the *Temporal Constraint Network* (TCN) [5]. A TCN is a constraint graph $< V, E >$ where the nodes V represent time-points (i.e. instantaneous events associated with the start or end of actions), while the edges E represent binary constraints on the duration between two time-points $x_i, x_j \in V$ of the form:

$$(l_1 \leq x_j - x_i \leq u_1) \vee \ldots \vee (l_n \leq x_j - x_i \leq u_n)$$

where $l_1, \ldots, l_n, u_1, \ldots, u_n \in \Re$ are the lower and upper bounds of the constraint. In other

words the time from x_i to x_j must lie in one of the intervals $[l_1, u_1]$, $[l_2, u_2]$, ... $[l_n, u_n]$. For example, if x and y represent the start and end time-points of action A then the constraint $(5 \leq y - x \leq 10) \vee (20 \leq y - x \leq 25)$ specifies that the duration of A is between 5 and 10 time units (e.g. minutes) or between 20 and 25 time units.

An interesting case is when only one interval $[l, u]$ is allowed for each edge. This is the *Simple Temporal Problem* (STP) restriction, whose graph counterpart is the *Simple Temporal Network* (STN). Checking the consistency of a TCN is known to be NP-complete, while in an STN, consistency can be determined in polynomial time, for instance using a local path-consistency propagation algorithm [9].

On the other hand, TCSPs may be generalized by allowing non-binary constraints. This is the *Disjunctive Temporal Problem* (DTP) [21], which formally is a constraint-satisfaction problem $< V, E >$ such that the constraints E are arbitrary disjunctions of STP-like constraints, i.e. each member of E has the form

$$(l_1 \leq x_{i_1} - x_{j_1} \leq u_1) \vee \ldots \vee (l_n \leq x_{i_k} - x_{j_k} \leq u_n)$$

where again $l_1, \ldots, l_n, u_1, \ldots, u_n \in \Re$. DTPs are thus conjunctions of n-ary disjunctive constraints: $\wedge_i(\vee_j c_{ij})$, where the c_{ij} are of the form $l \leq x - y \leq u$. As such, they can represent any other formula that we can construct with propositions c_{ij} (e.g. $c_{11} \Rightarrow c_{12}$).[1] The most frequently used way to solve a DTP is to convert it to a problem of selecting one disjunct, $(l_k \leq x_i - x_j \leq u_k)$ from each disjunctive constraint, such that the set of selected disjuncts forms a consistent STP. The DTP is consistent if and only if at least one of such *component STPs* is consistent. Checking the consistency of a DTP is NP-hard, and because DTPs include non-binary constraints, it is difficult to use path-consistency on them to increase efficiency [3]. However, several recent papers have presented heuristic techniques that significantly decrease the typical time needed to check DTP consistency [1, 13, 21, 22].

Recently, STPs have also been extended to take into account temporal uncertainty. In STPs (as in TCSPs and DTPs), the constraint between any two time-points x and y specifies an interval that is controllable: i.e. the execution agent can choose for $(y - x)$ any of the values within the allowed bounds given by the constraint. In realistic planning applications, however, the durations of some tasks or the time of occurrence of some events may depend on external parameters, and thus the actual value can only be *observed* by the agent at execution time (e.g., [11]). Such *contingent* values may be seen as being assigned by Nature while the other values are assigned by the executing agent. A *Simple Temporal Problem/Network with Uncertainty* (STPU/STNU) [26] is similar to a STP/STN except that the edges are divided into *contingent* and *requirement* edges. The finishing time-point of contingent intervals are controlled by Nature and hence *observed*, while others are controlled and hence *executed* by the agent. We will refer to this model later in the paper, as there is a strong relationship between consistency in an STNU (called controllability in the STNU setting) and consistency as we define it for CTPs.

3. Conditional Temporal Problems

We now introduce the Conditional Temporal Problem (CTP) formalism. Both in this section and throughout the remainder of the paper, we will illustrate our approach with a

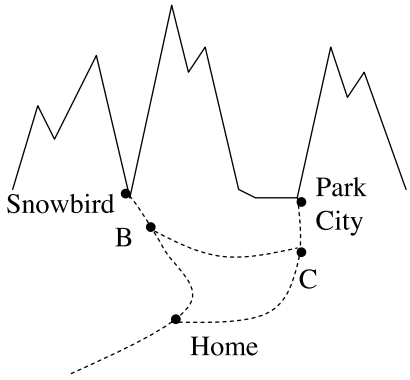

Figure 1. The map for the CNLP plan.

simple example derived from the one used in the original CNLP conditional-planning paper [15]. The example models a plan for going skiing, either at Snowbird or at Park City, starting from home, and traversing the roads (shown as dashed lines) depicted in Figure 1. As in the CNLP paper, it is possible that the road from point B to Snowbird and/or the road from point C to Park City will be covered with snow and impassable. However, the condition of those roads can only be observed from points B and C, respectively. For the purposes of the current paper, we also introduce two temporal constraints:

- If the agent skis at Snowbird, it should arrive after 1 p.m., because they offer great discounts in the afternoons.
- If the agent skis at Park City instead, it should arrive at point C no later than 11 a.m., because traffic is very heavy afterwards.

In our formalism, we will denote propositions with capital letters from the beginning of the alphabet, e.g. A, B, C. We will denote literals as A, or $\neg A$, and conjunctions of literals as lists of literals, e.g. $AB\neg C$ denotes $A \wedge B \wedge \neg C$.

Definition 3.1. A **label** l is a conjunction of literals, e.g. $l = ABC$.

Definition 3.2. Two labels l_1, l_2 are **inconsistent** (denoted as $Inc(l_1, l_2)$) iff $l_1 \wedge l_2$ is *False*. E.g. $l_1 = AB$ and $l_2 = \neg B\neg C$ are inconsistent. Two labels l_1, l_2 are **consistent** (denoted $Con(l_1, l_2)$) iff $\neg Inc(l_1, l_2)$). A label l_1 **subsumes** label l_2, (equivalently, l_1 is more specific than l_2), iff $l_1 \Rightarrow l_2$. E.g. if $l_1 = AB\neg C$ and $l_2 = A\neg C$, then l_1 subsumes l_2 (denoted $Sub(l_1, l_2)$).

Definition 3.3. The set of all possible labels defined with respect to a set of propositions P, is the **label universe** of P, P^*.

We are now ready to formally define Conditional Temporal Problems, inspired by the definitions of the TCSP, STP and DTP.

Definition 3.4. A **Conditional Temporal Problem (CTP)** is a tuple $< V, E, L, OV, O, P >$, where:

P is a finite set of propositions A, B, C, ...
V is a set of nodes (interchangeably called variables, time-points, events) {x, y, z, ...}
E is a set of constraints between nodes in V
$L : V \rightarrow P^*$ is a function attaching a label to each node, e.g. $L(n) = AB$
$OV \subseteq V$ is the set of observation nodes
$O : P \rightarrow OV$ is a bijection associating a proposition with an observation node. $O(A)$ is the node that provides the truth-value for proposition A.

CTPs can be interpreted as follows. When a value is assigned to a time point in V, it indicates the time of occurrence of that event. In planning problems, the time of occurrence is the time at which an action is executed.[2] The times assigned to the nodes in a CTP must satisfy the constraints in E. A major difference from other non-conditional temporal problems is that a node $v \in V$ should only be executed if its label's value becomes *True*. At the time of their execution the observation nodes provide the truth-value of propositions, which in turn determine the truth-value of labels. Note that each proposition has an implicit time point associated with it. Suppose that A is a fluent whose value may change over time (for example `battery-level-low`), and further suppose that the value of A needs to be observed at two different times. Then the CTP will include two distinct propositions, e.g. A_{T_1} and A_{T_2}, each associated by the function O to a different observation node. Therefore for each proposition in P there exists a unique observation node.

It is reasonable to assume that in any well-defined CTP there should not be any constraint relating nodes with inconsistent labels, since those nodes will never be executed under the same circumstances. Also, it is reasonable to require that when we execute a node v, we have to know the truth-value of its label $L(v)$. This in turn implies that (i) all nodes observing a proposition in $L(v)$ are executed in all cases in which v is executed, and (ii) they are executed before v. To ensure these requirements for any node v for which A appears in $L(v)$, we can statically check that $Sub(L(v), L(O(A))$ and add the constraint $O(A) < v$ to the temporal problem definition.

We now define three types of CTPs, which differ in the types of constraints they allow.

Definition 3.5. A *Conditional Simple Temporal Problem* **(CSTP)** is a CTP where the constraints in E are STP-like constraints, i.e. binary constraints (called edges) of the form $(l \leq y - x \leq u)$. We similarly define *Conditional Disjunctive Temporal Problems* **(CDTP)** and *Conditional Temporal Constraint Satisfaction Problems* **(CTCSP)**, by analogy to DTPs and TCSPs.

Example 3.1. Figure 2 shows a CSTP that encodes part of the ski-trip example. (To keep the example small, we omit the part of the plan that involves traveling from point C to Park City.) The vertices V represent three types of events:

– The start and end points of the *go* actions; (go x y) denotes the action of going from location x to location y. A node labeled $(\text{go x y})_S$, i.e. with subscript S, indicates the event of starting the action (go x y); similarly $(\text{go x y})_E$ denotes the event of ending such an action (and arriving at y).

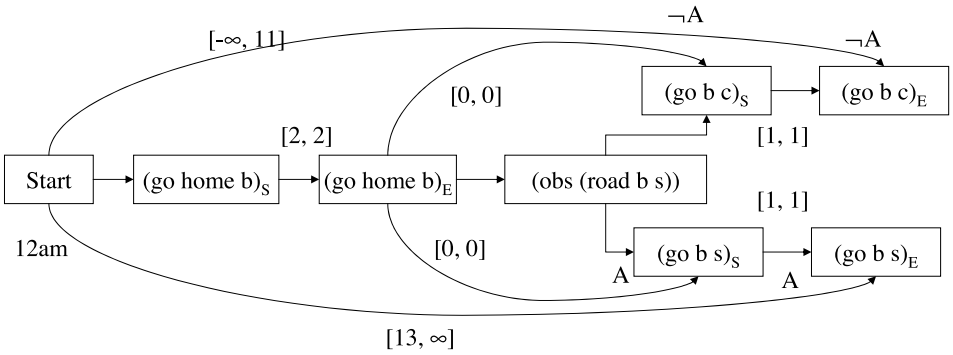

Figure 2. The skiing plan as a Conditional Simple Temporal Problem. All non-annotated edges are assumed [0, ∞] and all non-annotated nodes are assumed labeled *True*.

- The observation events; (obs (road x y)) denotes observing the condition of the road from *x* to *y*, while located at *x*. For clarity of presentation, we treat observation steps in our examples as if they were instantaneous, although in general this is not a requirement.
- The special *Start* event, which is associated with a specific arbitrary point in time: in our example, 12 a.m. It is used to encode absolute time constraints, for example, that if the agent goes to Snowbird, he should not arrive there until 1 p.m. (13 hours after *Start*). In the temporal-reasoning literature, the Start node is usually denoted "TR," for Temporal Reference point.

We will follow the convention that all non-annotated edges in the figures are assumed to have bounds [0, ∞] and the labels of the unlabeled nodes are *True*. There is exactly one observation action (obs (road b s)) that provides the truth-value of A, namely whether the road from b to s is open: thus, $O(A) =$ (obs (road b s)). Notice that this node satisfies the two conditions identified above: it is executed in every context in which steps labeled with A or $\neg A$ are executed (in fact, in this example, it is *always* executed), and it is executed before any of the steps that are labeled with A or $\neg A$ (note the implicit [0, ∞] constraint on the outgoing arcs from it).

Definition 3.6. An **execution scenario** s is a label that partitions the nodes in V into two sets V_1 and $V_2 : V_1 = \{n \in V : Sub(s, L(n))\}$ and $V_2 = \{n \in V : Inc(s, L(n))\}$, i.e. the set of nodes that will be executed because s implies their label is *True*, and the set of nodes that will not be executed because s implies their label is *False*. An execution scenario contains all the information (i.e. the value of all necessary propositions) to decide which nodes to execute and which not to execute. We will use **SC** to denote the set of scenarios for a given CTP.

Theorem 3.1 *Any complete truth-assignment to the propositions in P is an execution scenario (we will call it a* **complete scenario***). (See Appendix for proof).*

Definition 3.7. A **scenario projection** of the CTP $< V, E, L, OV, O, P >$ of an execution scenario s, denoted as **Pr(s)**, is the temporal non-conditional problem $< V_1, E_1 >$, where

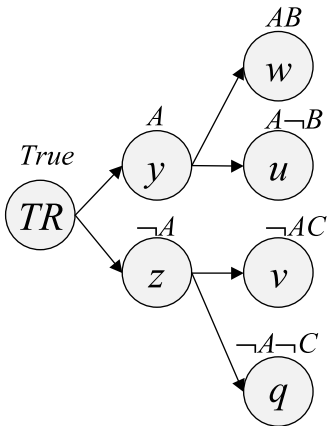

Figure 3. The CTP of the Example 3.2.

$V_1 = \{n \in V : Sub(s, L(n))\}$ and $E_1 = \{(\upsilon_1, \upsilon_2) \in E : \upsilon_1, \upsilon_2 \in V_1\}$. This will be an STP, DTP, or TCSP depending on the constraints in E. Put simply, the scenario projection of an execution scenario s is the set of nodes of the CTP that will be executed under s and all the constraints among them.

Example 3.2. In our skiing example, there are only two possible execution scenarios: A and $\neg A$, where **Pr(A)** includes, intuitively, all the nodes that are executed when the road to Snowbird is clear and the agent skis there, and, similarly, **Pr(¬A)** includes all the nodes that are executed when the road to Snowbird is closed and the agent instead heads towards Park City via point C. For a more interesting example, consider the CSTP in Figure 3, which has nodes $\{TR, y, z, w, u, \upsilon, q\}$ with labels $\{True, A, \neg A, AB, A\neg B, \neg AC, \neg A\neg C\}$, respectively. Suppose that $TR = O(A)$ observes A, $y = O(B)$ observes B, and $z = O(C)$ observes C. The CSTP thus has the typical structure of a conditional plan in which TR is initially executed and A is observed; if it is true y is executed, otherwise z is executed; and so on. The execution scenarios AB, ABC and $AB\neg C$ all refer to the same execution, i.e. under each of them, the nodes TR, y, and w, but no other nodes, will be executed.

Definition 3.8. Two execution scenarios are **equivalent execution scenarios** if they induce the same partition on the set of nodes. The "equivalent execution scenarios" relation induces an equivalence class relation **R**. A class in **R** contains all scenarios that are equivalent.

Definition 3.9. A scenario is a **minimum execution scenario** if it contains the minimum number of propositions compared to all other scenarios in its "equivalent execution scenario" class.

In Example 3.2 above, scenarios ABC, $AB\neg C$, and AB all belong to the same equivalence class, with AB being the minimum execution scenario of the class. For this example, there are four minimum scenarios: $\{AB, A\neg B, \neg AC, \neg A\neg C\}$.

4. Notions of Consistency

CTPs have a different notion of consistency than their non-conditional counterparts TCSPs, DTPs, STPs, and most other temporal reasoning problems. In the conditional case, consistency cannot be defined simply as the existence of an assignment to the time-points (nodes) that satisfies the constraints. This interpretation fails to take account of the fact that in the CTP modeling a temporal plan, propositions are usually only observed at execution time. We thus present three notions of consistency, which differ from one another in the assumptions made about when observation information is known.

In the first case, we require a notion of consistency that allows for off-line scheduling, i.e. allows precise times for all events to be determined before execution begins. Here it is reasonable to assume that the scheduling algorithm has no information about the outcome of the observations. Therefore it should schedule the nodes in such a way that the constraints are satisfied no matter how the observations turn out. If such a schedule exists, then we will say that the CTP is Strongly Consistent.

As a second case, consider an agent that plans for a number of initial future states. Each initial state corresponds to a set of initial truth-values of some propositions, i.e. an execution scenario. The specific scenario is unknown at planning time, but it will be known to the execution agent prior to execution. Then, it is necessary for the planning agent to verify that no matter which initial state turns out *True*, the execution agent has a way to execute the plan. If this is possible, the CTP is Weakly Consistent.

The third, most complicated and typical case, assumes that information about the outcome of observations becomes known during execution. Unlike the first case, however, it allows the decisions about the timing of events to be made dynamically at execution time. We will call a CTP Dynamically Consistent if it can be executed so that no matter what the outcome is for the upcoming observations, the current partial solution (i.e. the assignment of values to time-points) can be extended so that all constraints are satisfied.

These notions of consistency are similar to those developed for STPUs, where consistency is defined in terms of *controllability* [26]. Essentially, a network is controllable if there is a strategy for executing the timepoints under the agent's control that satisfies all requirements. Three primary levels of controllability had also been identified. In *Strong Controllability*, there is a static control strategy that is guaranteed to work in all situations. In *Weak Controllability*, for all situations there is a ''clairvoyant'' strategy that works if all uncertain durations are known when the network is executed. And in *Dynamic Controllability*, it is assumed that each uncertain duration becomes known (is observed) just after it has finished, and it requires that an execution strategy depend only on the past outcomes. Strong and Dynamic Controllability have been shown to be tractable [10, 26], while Weak Controllability is conjectured to be co-NP-complete [26].

Definition 4.1. A **schedule** T of a CTP $< V, E, L, OV, O, P >$ is a mapping $V \rightarrow \Re$ i.e. a time assignment to the nodes in V. We denote with $T(v)$ the time assigned to node v.

Definition 4.2. An **execution strategy** St is a function from the set of scenarios for a CTP to a schedule $St : \mathbf{SC} \rightarrow T$. A **viable execution strategy** is one such that $St(s)$ is a solution to the projection **Pr(s)** for each scenario $s \in \mathbf{SC}$.

Definition 4.3. Given a scenario s and a schedule T, for each node x in the projection scenario of s, we can determine the set of all observations performed before time $T(x)$, along with their outcomes. We will call the set of the observation outcomes before time $T(x)$ the **observation history** of x relative to scenario s and schedule T and will denote it as $H(x, s, T)$.

Definition 4.4. A CTP $< V, E, L, OV, O, P >$ is **Weakly Consistent** if there exists a viable execution strategy for it, i.e. every projection **Pr(s)** is consistent in the STP/DTP/TCSP sense.

Definition 4.5. A CTP $< V, E, L, OV, O, P >$ is **Strongly Consistent** if there is viable execution strategy St such that, for every pair of scenarios s_1 and s_2, and variable x executed in both scenarios,

$$[St(s_1)](x) = [St(s_2)](x)$$

Thus, an execution strategy that satisfies the definition of Strong Consistency assigns a fixed time to each executable timepoint irrespective of the observation outcomes. The idea behind finding a single schedule is similar to that of finding a conformant plan [7].

Definition 4.6. A CTP $< V, E, L, OV, O, P >$ is **Dynamically Consistent** if there is a viable execution strategy St such that, for every variable x and pair of scenarios s_1 and s_2,

$$Con(s_2, H(x, s_1, St(s_1))) \vee Con(s_1, H(x, s_2, St(s_2))) \Rightarrow [St(s_1)](x) = [St(s_2)](x)$$

In the definition above,[3] $Con(s_2, H(x, s_1, St(s_1)))$ means that the set of observation outcomes uncovered before x in scenario s_1 forms a label that is still consistent with scenario s_2 at the time at which x is to be performed in s_1. Thus, at time point x, the agent has not yet distinguished between scenarios s_1 and s_2. Therefore it must assign to x the same time in s_1 and s_2. The same arises in the opposite case (x in scenario s_2 while s_1 is still feasible). An execution strategy that satisfies the above definition ensures that the scheduling decisions that are taken (while executing) only use information available from previous observations.

To compare the three notions of consistency, reconsider the CSTP of Figure 2 as defined in Example 3.1. Is the network Strongly Consistent? We can immediately see that, if A is *True* (i.e. the agent observes that the road to Snowbird is open), then it must arrive and then immediately leave point b no sooner than noon, i.e. $(\text{go home b})_E \geq 12$, given the constraints $13 \leq (\text{go b s})_E - Start \leq \infty$, $0 \leq (\text{go b s})_S - (\text{go home b})_E \leq 0$, and $1 \leq (\text{go b s})_E - (\text{go b s})_S \leq 1$. That is, because the agent must not arrive in Snowbird before 1p.m., there is no place to wait at point b, and it takes an hour to get from b to Snowbird, it must arrive at b no earlier than noon. An analogous argument lets us deduce that if A is *False* the agent must arrive at b no later than 10 a.m., i.e. $(\text{go home b})_E \leq 10$ Thus, there is no way to construct a schedule for this network without knowing the truth-value of A. Hence, the CTP is not Strongly Consistent.

However, it is Weakly Consistent. To prove this we just have to provide a consistent schedule for each scenario. In this example there are two execution scenarios, $s_1 = A$ and

$s_2 = \neg A$, which means only the truth-value of A discriminates between possible scenarios. One consistent schedule for s_1, which we will call T_1, assigns (go home b)$_S$ a time of 10 a.m. (with all the other time points being directly derivable from that, e.g., (go home b)$_E$ is assigned noon). A consistent schedule for s_2, T_2 assigns (go home b)$_S$ a time of 8 a.m. The execution strategy St, where $St(s_1) = T_1$ and $St(s_2) = T_2$ is viable. So, provided only that the value for A is known before execution starts, the agent simply needs to pick the corresponding schedule T_1 or T_2.

Is the network Dynamically Consistent? In the discussion above we showed that T((go home b)$_E$) must be greater than or equal to 12 if A is observed *True*, and less than or equal to 10 if A is observed to be *False*. In turn, this forces T((go home b)$_S$) to be greater than or equal to 10 if A is *True*, and less than or equal to 8 if A is *False*. If we could observe A before starting out on the journey, i.e. before event (go home b)$_S$, then we could distinguish between the two scenarios, determine which one we fall into, and schedule our departure from home accordingly. However, the problem is set up such that being at point b is a precondition for the observation action. Thus, there is no way to perform the observation, and determine the value of A, ''in time'' to schedule the departure. The example is not dynamically consistent.

Let us now state an obvious property of the three notions of consistency that is similar to the corresponding one in STPU:

Theorem 4.1 *Strong Consistency* $\Rightarrow$ *Dynamic Consistency* $\Rightarrow$ *Weak Consistency.*

5. Strong Consistency

We now present an important property of Strong Consistency for CTPs.

Theorem 5.1 *A CTP* $< V, L, E, OV, O, P >$ *is Strongly Consistent if and only if the (non-conditional) temporal problem* $< V, E >$ *is consistent. (See Appendix for proof).*

The implication of the above theorem is that we can perform Strong Consistency checking by using specialized algorithms for non-conditional temporal problems such as IDCP [4] for STPs, known techniques [18] for TCSPs, and Epilitis [22] for DTPs.

5.1. Uses of the Strong Consistency Concept for Planning

A plan that is represented as a strongly consistent CTP can be executed according to a fixed schedule, in the sense that every action it includes has an assigned, specific time. Not all of the actions will occur in every scenario. Thus, the plan is contingent in the same sense as plans generated by CNLP [15]: Certain actions may or may not be executed, depending on the results of execution-time observations. A contingent plan with a fixed schedule should be contrasted with temporally contingent plan, in which the times at which actions are performed also depends upon such observations. Plans with necessary temporal contingencies will be dynamically consistent (as discussed in Section 7) but not strongly consistent.

Thus, Strong Consistency is a restrictive type of consistency, meaning that a CTP might not be Strongly Consistent but nonetheless can be executed. Nevertheless, since we can

employ existing algorithms and systems for determining Strong Consistency it is possible to build a temporal and conditional planner using those systems. Such a planner performs a search in the plan space adding appropriate actions, observations and temporal constraints to resolve the conflicts (threats) in the CNLP style. The consistency of the underlying temporal constraints in the CTP representing the current plan can then be determined.

6. Weak Consistency

6.1. Weak Consistency Checking

It is easy to design a brute force algorithm for checking Weak Consistency. The task involves finding a solution to the temporal subproblem $\mathbf{Pr(s_i)}$ for every execution scenario s_i. Thus, the two steps of the algorithm are:

1. Find the set of execution scenarios **SC**.
2. Check the consistency of the non-conditional problem **Pr(s)**, $\forall s \in \mathbf{SC}$.

Let us examine a specific example of the above, on the CSTP of Figure 2. The two projections **Pr(A)** and **Pr(** $\neg$**A)** and all the constraints in these two projections are shown in Figure 4. Consistency can be easily proven in each projection.

We can improve on this algorithm by noticing that $\mathbf{Pr(s_i)} = \mathbf{Pr(s_j)}$ for every two equivalent scenarios s_i and s_j. Thus, we only need to select one scenario s_i from each class in **R** of Definition 3.8 and check the consistency of $\mathbf{Pr(s_i)}$. It might even be desirable to select the minimum execution scenario as the representative of its class. Then the first step of the algorithm is the problem of finding the *set of minimum execution scenarios*. We solve this problem in [22] where we present an algorithm that calculates this set without explicitly enumerating all possible scenarios. We also prove the complexity result that Weak CSTP Consistency is co-NP-Complete (see Theorem A.3 in the Appendix).

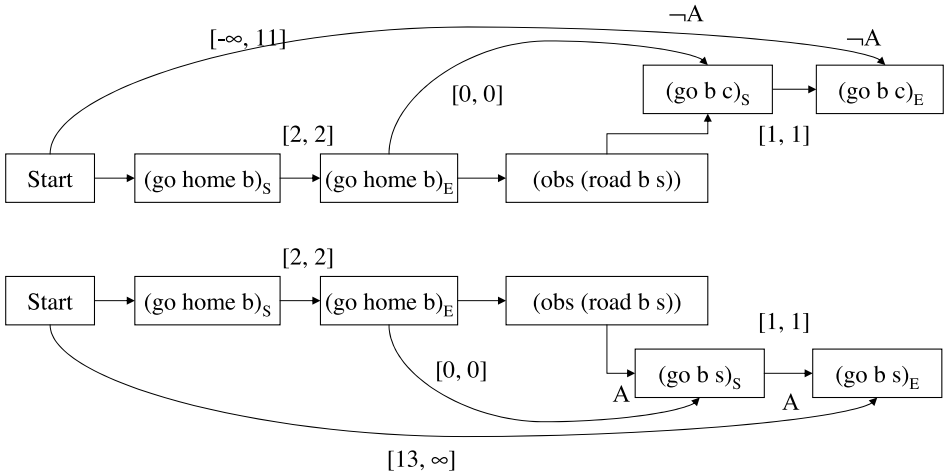

Figure 4. The projected STPs of Figure 2 for all scenarios. **Pr(**$\neg$**A)** is shown in the top part and **Pr(A)** in the bottom part.

Another way to improve the Weak Consistency checking algorithm is to perform the second step of the algorithm incrementally. For example, when solving the sequence of problems $\mathbf{Pr(s_1)}$, $\mathbf{Pr(s_2)}$, ... there is often shared computation between problems $\mathbf{Pr(s_i)}$ and $\mathbf{Pr(s_{i+1})}$. The order of consideration of each **Pr(s)** highly influences the amount of computation that can be shared. An algorithm that employs this idea and calculates an appropriate order of incremental consistent checks for the series of $\mathbf{Pr(s_i)}$ is again presented in [22] (Chapter 6) for the CSTP case.

6.2. Uses of the Weak Consistency Concept for Planning

If a plan is represented as a weakly consistent CTP that is not also dynamically or strongly consistent, this means that there is always a non-contingent plan for any set of observations, but to execute that plan, we must know the observations at the start of execution.

We now suggest a possible use of Weak Consistency for planning purposes, namely in planning architectures that are based on plan-merging. Examples of such architectures are Workflow Management Systems [6], PRS [12], the Plan Management Agent [25], and Autominder [16] to name a few. In these systems, there is a library of plan (or workflow) schemata, and whenever a new goal arrives, a plan from the plan library is selected and subsequently merged with the system's existing commitment structure, i.e. the set of (partially) instantiated plans which it has already adopted. The plan to be merged in the context will depend both upon the new goal to satisfy and the current set of commitments. The conditions set by the latter form a "scenario" for which there should exist a corresponding schedule, which makes Weak Consistency relevant. The plan library may include a number of different schemata that achieve a given goal G. A CTP can be used to compactly represent all such schemata. For example, if P_1 achieves goal G when A is *True*, and plan P_2 achieves G when $\neg A$, then we can build a CTP that simultaneously represents both P_1 and P_2 by attaching appropriate labels A and $\neg A$ on the temporal variables in the plans. Now assume that P_1 and P_2 share a large part of their structure and differ only in a few preparatory steps. It is easy to see that the CTP representation can attach the label *True* to all common steps and this way both display the shared structure and remove the redundancy.

As a simple example consider the CTP shown in Figure 5(a), which encodes the four non-conditional plans in Figure 5(b). (Imagine that label A denotes "fuel tank empty" and label B denotes "load over 2000 lbs"). Not only is the CTP encoding more compact, but checking its consistency using efficient Weak Consistent checking algorithms will be faster, by definition, than individually checking the consistency of each of the non-conditional temporal plans.

7. Dynamic Consistency

7.1. Dynamic Consistency Checking

We now turn to Dynamic Consistency. In order to distinguish between the execution of the same node in different scenario projections, we use $N(x, s)$ to denote node x in $Pr(s)$. Also, let us denote with $Diff_{s_2}(s_1)$ the set $\{N(v, s_1)\}$ of all nodes v in s_1 that provide observations with outcomes that differ from the corresponding ones in s_2.

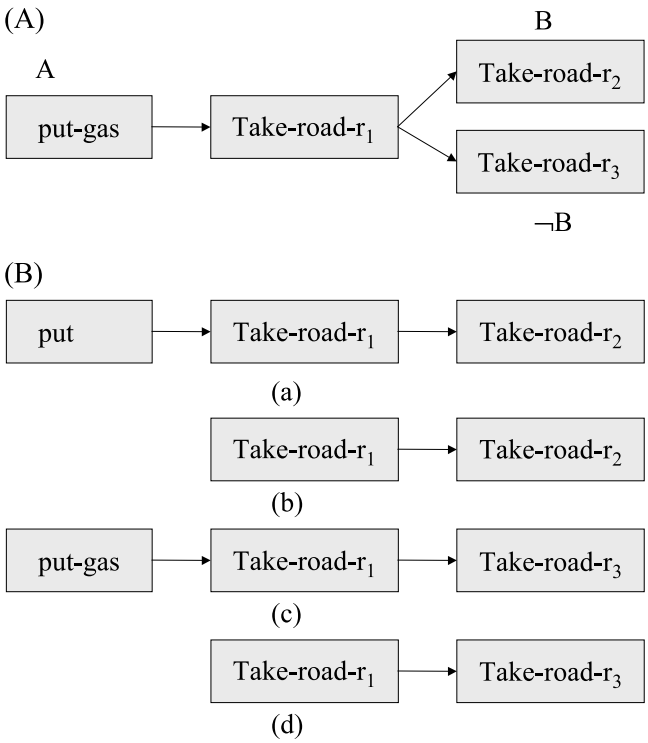

Figure 5. (A) A CTP encoding four different plan schemata. (B) The non-conditional plan schemata represented by the CTP in (A).

To prove Dynamic Consistency of a CTP we need to identify a viable execution strategy *St* satisfying the conditions of Definition 4.6. The condition $Con(s_2, H(x, s_1, St(s_1)))$ is satisfied if and only if at the time $N(x, s_1)$ is executed, there is no observation node $N(v, s_1)$ in $Diff_{s_2}(s_1)$ that has been executed yet; otherwise, we could distinguish between the two scenarios. This directly translates to the following equation for $Con(s_2, H(x, s_1, St(s_1)))$:

$$Con(s_2, H(x, s_1, St(s_1))) \Leftrightarrow \bigwedge_{N(v,s_1)\in Diff_{s_2}(s_1)} N(x, s_1) \leq N(v, s_1)$$

and similarly for $Con(s_1, H(x, s_2, St(s_2)))$. Thus, we can rewrite the condition in Definition 4.6 as:

$$\begin{aligned}\{\bigwedge_{N(v,s_1)\in Diff_{s_2}(s_1)} N(x, s_1) \leq N(v, s_1)\}\\ \bigvee\{\bigwedge_{N(v,s_2)\in Diff_{s_1}(s_2)} N(x, s_2) \leq N(v, s_2)\}\\ \Rightarrow N(x, s_1) = N(x, s_2)\end{aligned} \quad (1)$$

On the left-hand side of the implication, we have simply replaced each of the two *Con* conditions from Definition 4.6 with a conjunction over times of observation nodes; the right-hand side of the implication has remained unchanged.

DynConsistency(CTP *Ctp)*
1. DTP $D :< V,C >=< \cup_i V_i, \cup_i E_i >$,
   where $Pr(s_i) =< V_i, E_i >$ are the projected scenarios.
2. For each pair of scenarios s_1, s_2
3. For each node v that appears in both s_1, s_2
4. $C = C \wedge DC(v, s_1, s_2)$
5. EndFor
6. EndFor
7. If D is consistent, return Dynamic-Consistent
8. Else return non-Dynamic-Consistent

Figure 6. The dynamic consistency algorithm.

The main idea behind the consistency checking algorithm is to view the above condition of the definition as a (disjunctive) constraint between nodes $N(x, s)$: These together with the set of all nodes $N(x, s)$ for every node x and scenario s of the original CTP define a new temporal problem, namely a DTP D. With this reformulation, an execution strategy St defines a schedule T of D and vice versa by setting $[St(s)](x) = T(N(x, s))$. The aim is to add appropriate constraints to D so that a solution schedule of D will correspond to a dynamic execution strategy and vice versa. So, in addition to the constraints resulting from Equation (1), we need to impose on the nodes of D all the constraints in every projection **Pr(s)**. Then, every solution to D will satisfy both the constraints in each projection (thereby guaranteeing that the corresponding strategy will be viable), and the Dynamic Consistency conditions. These ideas lead to the design of the algorithm in Figure 6, where **DC(x, s_1, s_2)** (called a DC constraint) is used as a shorthand of Equation (1) above.

Let us trace the algorithm on a specific example such as the CSTP of Figure 2 where $O(A) =$ (obs (road b s)). Line 1 of the algorithm creates a DTP with all the nodes and edges in the two projections **Pr(A)** and **Pr(¬A)**. The result is shown in Figure 7 (with

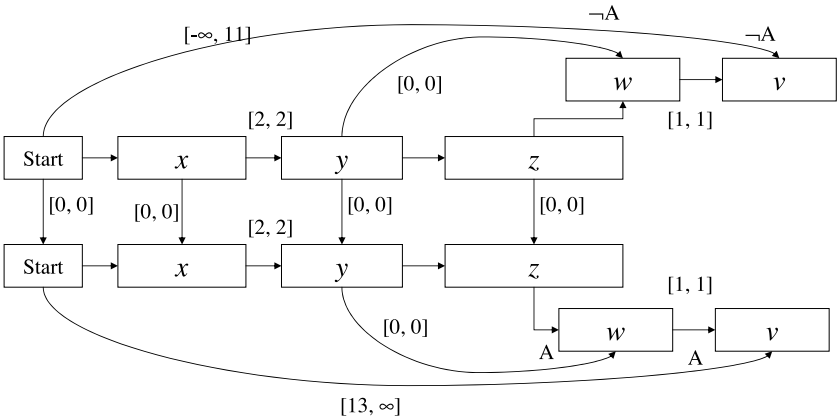

Figure 7. The DTP created by the Dynamic Consistency algorithm, including **Pr(¬A)** (top part), **Pr(A)** (bottom part), and DC constraints between them.

some additional constraints explained below). To simplify the equations we renamed the nodes (go home b)$_S$, (go home b)$_E$, (obs (road b s)), (go b s)$_S$, and (go b s)$_E$ as x, y, z, w, and v. We notice that $Diff_{\neg A}(A) = \{N(z, A)\}$ and $Diff_A(\neg A) = \{N(z, \neg A)\}$ and so the DC for *Start*, $DC(Start, A, \neg A)$, is

$$N(Start, A) \leq N(z, A) \vee N(Start, \neg A) \leq N(z, A) \Rightarrow N(Start, A) = N(Start, \neg A).$$

Since *Start* is before z in both cases, $N(Start, A) = N(Start, \neg A)$. Similarly, we find that $N(x, A) = N(x, \neg A)$, $N(y, A) = N(y, \neg A)$, and $N(z, A) = N(z, \neg A)$. For nodes w and v the antecedent of the DC implication is always *False* (they occur after z in both scenarios) and so the DC constraint is already satisfied. The result after adding all the DC constraints in Line 4 of the algorithm is shown in Figure 7. The resulting DTP is actually an STP and it is inconsistent, indicating that the original CTP is not Dynamically Consistent.

Suppose instead that the constraints ordering z after y are dropped and z can now be executed any time after *Start*. In particular, other constraints permitting, it could be scheduled before x and y. Then, the DC constraint for x specifies that either x occurs before z in which case $N(x, A) = N(x, \neg A)$, **or** it occurs after z in both scenarios. Thus, the DC constraints are disjunctive in general (recall that $a \Rightarrow b$ is equivalent to $\neg a \vee b$). In the case where z is allowed to be executed before x and y the CTP is Dynamically Consistent. Semantically this corresponds to the case where we observe whether the road from b to s is open before we leave home. In that case, we can decide when to start the trip for each different scenario.

Notice that in Equation (1) if the observation nodes in all scenarios are constrained to be ordered with respect to each other, then the conjunctions over all $N(v, s_i) \in Diff_{s_j}(s_i)$ can be substituted with the single minimum of the order. Then Equation (1) becomes

$$N(x, s_1) \leq N(n, s_1) \vee N(x, s_2) \leq N(m, s_2) \Rightarrow N(x, s_1) = N(x, s_2) \tag{2}$$

n and m being the nodes for which $N(n, s_1)$, $N(m, s_2)$ are minimum in the order of the nodes in $Diff_{s_2}(s_1)$ and $Diff_{s_1}(s_2)$ respectively. In general, observation nodes that are constrained to be scheduled after others in the sets $Diff_{s_2}(s_1)$ and $Diff_{s_1}(s_2)$ are ruled out of the conjunctions in Equation (1).

A more complicated example is shown in Figure 8 where two observation nodes $O(A) = x$ and $O(B) = y$ are unordered with each other so Equation (1) cannot be simplified to the form of Equation (2). Let us consider node x and scenarios $s_1 = AB$ and $s_2 = A\neg B$. Then, $Diff_{s_2}(s_1) = \{N(y, s_1)\}$ while $Diff_{s_1}(s_2) = \{N(y, s_2)\}$. Thus, $DC(x, s_1, s_2)$ is the constraint $N(x, s_1) \leq N(y, s_2) \vee N(x, s_2) \leq N(y, s_2) \Rightarrow N(x, s_1) = N(x, s_2)$. If we decide to perform the observation for A first, i.e. $x < y$, then $DC(x, s_1, s_2)$ becomes $N(x, s_1) = N(x, s_2)$. The resulting STP is shown in Figure 9(a). In the other case (where we defer the observation of A) we end up with the STP in Figure 9(b) where there is no constraint between $N(x, s_1)$ and $N(x, s_2)$. Since the observations are unordered, the DC constraints are disjunctive and represent in a DTP both of these alternative STPs of (a) and (b). The original CTP is Dynamically Consistent, if and only if one of these alternatives is consistent.

It is important to note that we reduced the consistency checking problem to a DTP because DTPs can represent n-ary disjunctive constraints. TCSPs and STPs do not allow

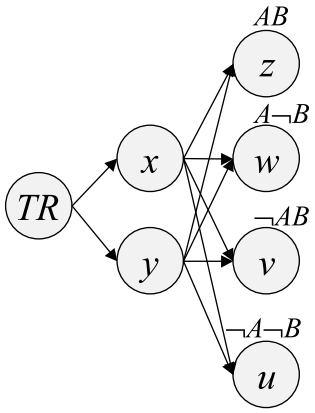

Figure 8. A CTP with two observation nodes unordered with respect to each other.

this, and thus would not satisfy our requirements. Constraints of this type are typical of Dynamic Consistency in CTPs and make the problem intractable in general. This contrasts with Dynamic Controllability in STPUs in which constraints can be reduced to simple STP constraints, hence allowing the design of polynomial-time solution algorithms. Thus, DTP solving algorithms such as Epilitis [22], which include a number of highly effective heuristic pruning techniques, will have a direct effect on Dynamic Consistency checking in CTPs.

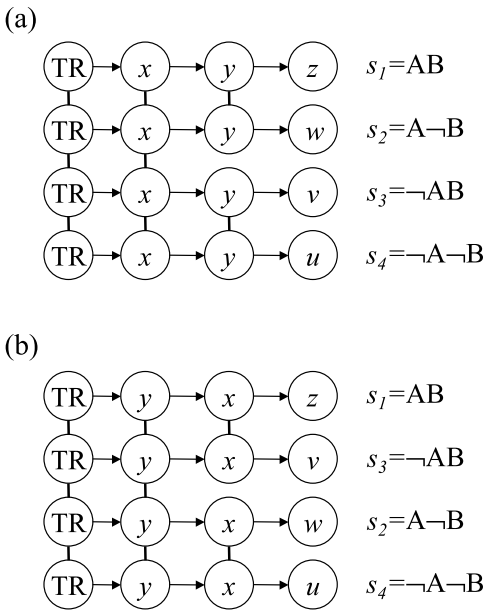

Figure 9. The STP projections with DC constraints for each order of observations. Directed edges are assumed $[0, \infty]$ and undirected edges denote [0, 0] constraints.

Regarding the complexity of the DTP D, notice that D contains $O(|V|\ |SC|)$ variables, where V is the set of variables in the CTP and SC the set of (minimal) scenarios whose number is in the worst case exponential to the number of propositions $|P|$. In the worst case the constraints are disjunctive and, when put in Conjunctive Normal Form, may create an exponential number of disjunctive clauses. Nevertheless, some structural properties of the CTP help in reducing the complexity.

First, as we have noted above, when the observation nodes are ordered with respect to each other in every scenario, the DC constraints are given by Equation (2), which is a great simplification over Equation (1). Additionally, if each node is ordered with respect to every observation node in all scenarios, then the antecedent of each DC constraint can be statically checked. In this case, the DC constraint either becomes $N(x, s_1) = N(x, s_2)$ or it is already satisfied. For instance a CSTP can then be made equivalent to a larger STP, since no disjunctive constraints are added to the problem, which allows very efficient Dynamic Consistency checking.

7.2. Uses of Dynamic Consistency Concept for Planning

Dynamic Consistency checking can be used to build a temporal and conditional planner. It is easy to see that by appropriately modifying the CNLP algorithm [15] it is possible to allow the simultaneous representation of and reasoning with quantitative temporal constraints and conditional branches. When a temporal constraint $x < y$ is added to the CTP representing the conditional plan (e.g. to resolve a conflict), the Dynamic Consistency algorithm can determine whether the resulting plan is executable. Moreover, this notion of "executable" goes beyond that of traditional planning systems, because it allows for observations to be made at execution time in plans in which timing constraints depend on observation outcomes. Dynamic Consistency checking can also support the merging of such richly expressive plans at execution time, e.g. to handle new goals that arise during execution [22] (Chapter 7).

In order to execute a Dynamically Consistent plan we can instead execute the DTP D to which we reduced the problem. Notice that we should execute only one node $N(x, s_i)$ for every scenario s_i since all such nodes semantically correspond to the same event and the same CTP node. Of course, the algorithm guarantees that $N(x, s_i) = N(x, s_j)$ in all appropriate cases and avoids confusion. We can identify at least three ways D can be executed: (i) We compute a solution to D and execute that. This is the least flexible approach since it commits to a specific schedule (solution) of D. (ii) We find and flexibly execute a consistent component STP of D. Consistent components STPs are returned by DTP solvers such as Epilitis [22] and can be flexibly executed with algorithms such as in [23]. (iii) We flexibly execute the DTP directly, retaining all possible scheduling flexibility, using the algorithm in [24].

Finally, we note that because typical conditional plans satisfy both of the conditions mentioned at the end of the previous subsection, the performance of the DC algorithm during plan construction and merging is likely to be higher than in the general case. In addition, conditional planners generate plans where the number of distinct execution scenarios is linear in the number of propositions. We suspect that in this case the Dynamic Consistency algorithm we presented is actually polynomial in the number of original CTP variables and propositions [22] (Chapter 6). We intend to formalize these ideas on performance improvements in our future work.

8. Improved Conditional Planning

In the previous section, we noted that by using a Dynamic Consistency algorithm, one can extend traditional conditional planners to support quantitative temporal constraints. It is essential to manage those constraints; if they are ignored, then the planner risks generating incorrect plans. For instance, a CNLP-style planner would generate a conditional plan for our skiing example if it simply ignored the two additional temporal constraints (either arrive at Snowbird after 1 p.m., or else arrive at point *C* on the way to Park City before 11 a.m.). But such a plan would be useless, because, as we have already seen, given the temporal constraints the plan is dynamically inconsistent and there is no way of executing it.

Of course, the traditional conditional planners [14, 15, 17] were not designed to deal with quantitative temporal constraints. But they do perform a limited form of temporal reasoning, in order to deal with ordering constraints, and it turns out that even for plans with only ordering constraints, there are clear advantages to using the dynamic consistency approach.

CNLP propagates context information only along causal links and conditioning links, but not along ordering constraints. We assume that this choice was made so that if a step x with context *True* is promoted after a step y of context A, then the context of x remains *True* and so x can be reused to provide causal links to steps in other contexts, thereby potentially reducing the amount of planning required and resulting in smaller plans.

Nevertheless, this method might reject valid (i.e. executable) plans. An example is shown in Figure 10(a). The bold edges correspond to causal links, and the lighter edges

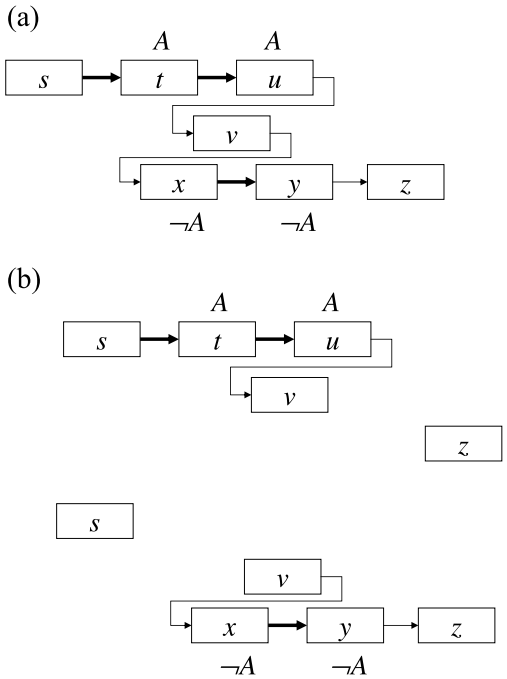

Figure 10. (a) A CNLP plan whose handling by CNLP reasoning falsely induces that s is necessarily before z. (b) The two projections of the plan: z is allowed before s in both.

denote ordering constraints added by threat resolution. We suppose that step v clobbers both the causal link $t \rightarrow u$ and the causal link $x \rightarrow y$; hence v has been promoted and demoted respectively to resolve the conficts. We further suppose that z clobbers the latter causal link and has been promoted after y.

When CNLP checks whether an ordering constraint exists between a pair of nodes s and z, it essentially computes the transitive closure and determines whether $s < z$ holds. Since the context information is ignored in this calculation, CNLP essentially calculates Strong Consistency of the induced CTP. However, in Figure 10(b) the two projections of the plan are shown, and it is easy to see that s and z are actually unordered with respect to each other.

As already mentioned in Section 5, Strong Consistency is a restrictive type of consistency and plans that are executable (i.e. Dynamically Consistent) might not be Strongly Consistent. Thus, *CNLP is not complete and it might reject valid plans*, unlike what is conjectured in the original CNLP paper. In the above example, if z is ordered before s and this is the only valid plan, CNLP will reject it as inconsistent even though it is Dynamically Consistent (assuming A is observed before this portion of the plan).

9. Related Approaches and Conclusions

As far as we know only two approaches might be compared to our work. The first, a paper by Schwalb et al. [19], separates propositional and temporal reasoning, addressing expressive propositional and temporal constraints, to process deduction and hypothetical reasoning on knowledge bases. The authors define a "Conditional Temporal Network" model, in which some constraints are dependent on a condition and are only used if that condition is *True*. The aim is to make queries in the base such as "is formula F consistent with the current constraints?" Although such a model may be seen as a general logical framework for doing conditional temporal reasoning, it is insufficient for our purpose for two reasons. First, the approach is static and does not deal with the dynamic aspects of plan execution: the time at which a condition is known to be *True* or *False* is not considered, which for planning purposes is crucial. Second, unlike Weak Consistency, which determines whether all scenarios are consistent, they determine whether there is at least one consistent scenario. This is sufficient when processing queries on a knowledge base, when one interpretation is searched for, but in our planning context that would only mean there exists one unique scenario in which the plan will not fail.

The second and more interesting paper is that of Barber [2], which combines quantitative temporal constraints and alternative contexts in a kind of networks that we will call BarN.[4] Barber defines a temporal problem where constraints (instead of nodes) are annotated with a label (in his terminology a context). Consistency in a BarN corresponds to Weak Consistency in a CTP.

A primary difference between BarNs and CTPs is thus that the former is based on conditional constraints while the latter is based on conditional events. In the Appendix (Theorem A.4) we show that we can use the latter to represent the former and thus our formalism is at least as general as Barber's. Because contexts in BarNs are associated with labels, not nodes, the translation from conditional planning is not as clear as with CTPs, which very naturally associate an observation with a node and attach appropriate labels to

subsequent nodes. CTPs can then readily check various forms of consistency, using the techniques described earlier. In contrast, with a BarN representation, the planner has to construct the context hierarchy itself given the observations. Perhaps the most important ramification of this implicit treatment of observations is that the notion of Dynamic Consistency cannot be defined for BarNs. This is because the truth value of the contexts is not associated with a particular time-point.

Unlike BarNs, our new Conditional Temporal Problem formalism is geared towards planning and execution purposes. It is a constraint-based formalism for temporal reasoning in the face of uncertain–or contingent–events, and we have described its usefulness for conditional planning. There are many avenues for future research, of which we highlight a few. CTPs deal with temporal uncertainty arising from the outcome of observations, while STPUs handle uncertainty regarding the timing of uncontrollable events. Obviously, a hybrid model and algorithms that handle both sources of uncertainty would be highly desirable. We are also working on identifying minimal structural requirements for CTPs that will enable polynomial-time Dynamic Consistency algorithms. In parallel, we are also investigating efficient Weak Consistency algorithms.

Acknowledgments

Work on this project by the first and third authors has been partially supported by the United States Air Force Office of Scientific Research (F49620-01-1-0066), by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Office, Air Force Materiel Command, USAF (F30602-00-2-0621), by the National Science Foundation (IIS-0085796), and by Andrew Mellon Predoctoral Fellowship. The views and conclusions herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR, DARPA, AFRL, or the U.S. Government. We would also like the anonymous reviewers for their constructive comments.

Appendix

Theorem A.1 *Any complete assignment to the propositions in P is an execution scenario.*

Proof: Let s be a complete truth-assignment to the propositions in P, l be a label and $p_1 \ldots p_n$ be the propositions that appear in l (either positive or negated). The set of p_i will also appear in s since s is complete. If any p_i appears with different sign in s and l then s and l are inconsistent. Otherwise, if all p_i appear with the same sign in s and l then s subsumes l. So, every node, no matter what its label is, will either belong to V_1 or V_2 in Definition 3.6. □

Theorem A.2 *A CTP $< V, L, E, OV, O, P >$ is Strongly Consistent if and only if the (non-conditional) temporal problem $< V, E >$ is consistent.*

Proof: The theorem states that we can determine Strong Consistency by ignoring the label and the observation information of the nodes in the CTP and just calculate consistency as we would for an STP, TCSP, or DTP depending on the kind of constraints in E.

"⇐" Suppose that the CTP is Strongly Consistent. Then we will show that the temporal problem $< V, E >$ is also consistent. Let T be a schedule of all nodes, such that $T(x) = [St(s_i)](x)$, where s_i some scenario where x is executed. $T(x)$ is a function because (i) x appears in at least one scenario (or it can be removed from the CTP), and (ii) a Strong execution strategy specifies a unique value to every $[St(s_i)](x)$ for all scenarios s_i where x appears. Because St is viable, T satisfies the constraints in all $\mathbf{Pr(s_i)} = < V_i, E_i >$, for every scenario s_i. Since T satisfies the constraints in every set E_i it satisfies the constraints in their union $\cup_i E_i = E$. Thus, T is a solution to $< V, E >$ and so the latter is consistent.

"⇒" Suppose that the temporal problem $< V, E >$ is consistent; we will prove that the CTP $< V, E, L, OV, O, P >$ is Strongly Consistent. Let T be any solution of $< V, E >$ (it has to have at least one since it is consistent). For every s_i, T is also a solution of $\mathbf{Pr(s_i)} = < V_i, E_i >$ (ignoring any irrelevant assignments $T(x)$ where x does not appear in V_i) since $E_i \subseteq E$. The execution strategy $St(s_i) = T$ is viable (since T is a solution to every $\mathbf{Pr(s_i)} = < V_i, E_i >$) and also $[St(s_i)](x) = [St(s_j)](x) = T(x)$, $\forall x$ as the definition of Strong Consistency requires. □

Theorem A.3 *Weak CSTP Consistency checking is co-NP-complete.*

Proof: We will prove the result by translating in polynomial time and space a SAT problem to the co-problem of checking Weak Consistency, the co-problem being finding a scenario s_i such that $\mathbf{Pr(s_i)}$ is inconsistent. Specifically, we will create a CSTP given a SAT problem such that the SAT problem has a solution, if and only if there is a scenario s_i such that $\mathbf{Pr(s_i)}$ is inconsistent.

Given the SAT problem with Boolean variables $B = \{x, \ldots, y\}$ and clauses of the form $C_i = (x \vee \ldots \vee y \vee \neg z \ldots \neg w)$, $i = 1 \ldots K$ we create a CSTP $< V, E, L, ON, O, P >$ as follows: The set of propositions is $P = B = \{x, \ldots, y\}$. For each clause $C_i = (x \vee \ldots \vee y \vee \neg z \vee \ldots \vee \neg w)$ and each variable appearance x or $\neg x$ in C_i we create a time-point X that we include in V, with label $L(X) = x$ or $L(X) = \neg x$ respectively. Let us denote with $Clause(C_i)$ the nodes of the CSTP that were included because of C_i. Since we are checking for Weak Consistency it does not matter which nodes are observation nodes. The last thing to define are the constraints between the nodes. There is an constraint between a variable $X \in Clause(C_i)$ to each variable Y with consistent label in $Clause(C_{(i+1)modK})$: $Y - X = -1$ (we will drop the $modK$ clause in the rest of the proof for clarity; just remember that the nodes in the last $Clause(C_i)$ are connected to the nodes in the first $Clause(C_1)$). The translation is obviously linear in the number of SAT variables and linear in the number of clauses of the SAT problem.

Figure 11 illustrates the proof concept. It presents an example, by showing the resulting CSTP from the SAT problem $(x \vee y \vee z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee \neg y \vee \neg z) \wedge (\neg y \vee z)$. The labels of each node appear on its top right corner. Notice that there are three propositions in the CSTP for the three SAT variables x, y, z that appear in the labels, either as positive or negative literals, and eleven CSTP nodes one for each appearance of a variable in any clause. The nodes in the CSTP are arranged in columns corresponding to $Clause(C_i)$, $i = 1 \ldots 4$ and are named with the variable of the corresponding proposition and the index i. The edges are connected from a node in $Clause(C_i)$ to all the nodes in $Clause(C_{i+1})$. The order of appearance of clauses in the SAT problem is arbitrary. Also recall that all the edges from a node X to a node Y correspond to the constraint $Y - X = -1$ not

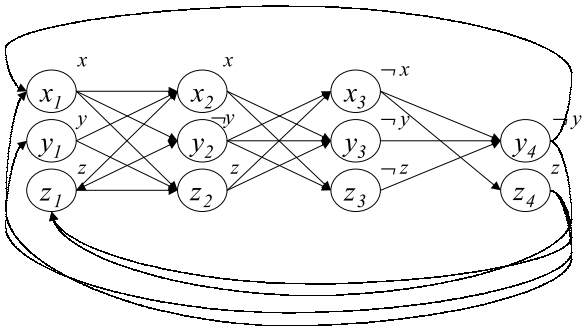

Figure 11. The CSTP resulting from translating a simple example TSAT problem.

shown in the figure for clarity. Notice also that there are no edges between nodes with inconsistent labels, e.g. x and $\neg x$.

Let us assume that the SAT problem has a solution $\{x = True, \ldots, y = True, z = False, \ldots, w = False\}$. Since this solution makes *True* at least one variable in a clause, it will make *True* the label of at least one CSTP time-point within *Clause*(C_i). We will call a time-point whose label becomes *True* in *Clause*(C_i) L_i (there may be more than one). Notice that each L_i has to have an edge to L_{i+1} because it cannot be the case that L_i has a label x and L_{i+1} a label $\neg x$ by the way the SAT solution is constructed (it never assigned x both *True* and *False* at the same time). Thus the set $\{L_i, i = 1 \ldots K\}$ forms a negative cycle (with weight $(-1) \times K$). There must be at least one scenario s that makes all the nodes in $\{L_i, i = 1 \ldots K\}$ *True*. Namely, let s be the complete scenario that corresponds to the SAT solution (s is a scenario by Theorem 3.1). In other words, **Pr(s)** is an inconsistent projected STP. In the above example, the SAT solution $\{x = True, y = False, z = True\}$ makes the labels of the CSTP nodes x_1, x_2, y_2, y_3, y_4, z_1, z_2, and z_4 *True* and all the rest *False*. The former set forms at least one negative cycle, e.g. $\{x_1, x_2, y_3, z_4\}$. This completes the proof that if the SAT problem has a solution, the CSTP is not Weakly Consistent.

We will now prove the converse, namely that if the SAT problem has no solution, then the CSTP is Weakly Consistent. Take any complete assignment to the SAT variables. Any such assignment also corresponds to a scenario of the CSTP (by Theorem 3.1). If SAT has no solution, for every such assignment/complete scenario s there is at least one clause C_i that is not *True*, or in other words, all the SAT literals of C_i have to be *False* too. Thus, all the time-points of *Clause*(C_i) are inconsistent (not executed) under scenario s. But, by the way the CSTP is constructed, every cycle (negative or not) has to go through all clauses. Since no time-point in *Clause*(C_i) becomes *True* under s, there can be no negative cycle in **Pr(s)** for any scenario s.

The above argument shows that checking Weak CSTP Consistency is co-NP-hard. Since checking if an STP is consistent is a polynomial problem, co-Weak Consistency is also in co-NP and thus the problem is co-NP-complete. □

Theorem A.4 *Every (conditional) constraint of the form* $l_1 \leq x_1 - y_1 \leq u_1 \vee \ldots \vee l_k \leq x_k - y_k \leq u_k$ *that should hold only when label l is True, can be represented with only conditional events.*

Proof: For any constraint that we want to represent of the form $l_1 \leq x - y \leq u_1 \vee l_2 \leq s - t \leq u_2$ with condition (label) l, we create the dummy nodes w, z, u, v all having label l. We then insert constraints requiring that the pairs of time-points $\{x, w\}$, $\{y, z\}$, $\{s, u\}$, and $\{t, v\}$ co-occur (e.g. $0 \leq x - w \leq 0$), and we also add the (unconditional) constraint $l_1 \leq w - z \leq u_1 \vee l_2 \leq u - v \leq u_2$. This way when l is *True*, nodes w, z, u and v will be executed and, because they co-occur, the original (conditional) constraint on nodes x, y, s and t will be imposed. □

Notes

1. Note that this is true despite the fact that DTPs do not include negated literals, because $\neg c_{ij} : l \leq x - y \leq u$ can always be rewritten as $x - y < l \vee x - y > u$, approximated as close as desired by $x - y \leq l - \epsilon \vee x - y \geq u + \epsilon$.
2. If the action has temporal duration, the model includes two nodes for every action: one that denotes the time at which execution of the action begins, and another that denotes the time at which it ends.
3. We point out that our consistency definitions and algorithms would still be valid had we defined labels as any propositional formula on P^*. We chose to restrict labels to conjunctions only for illustration purposes. Also, note that although they are symmetric, both disjuncts *Con*(s_2, *H*(x, s_1, *St*(s_1)) and *Con*(s_1, *H*(x, s_2, *St*(s_2))) are required for the definition to cover only the set of plans that are actually executable. Here is an example that shows that using only one of the disjuncts is not enough. Consider the CTP with nodes x, y, z, w, $L(z) = A$, $L(w) = \neg A$, $L(x) = L(y) = True$, and $O(A) = y$, and constraints $x - y \in [-5, 5]$, $z - y = 5$, $w - y = 15$, $z - x = 10$, and $w - x = 10$. If A is *True*, only the schedule $T_1(x) = 0$, $T_1(y) = 5$, and $T_1(z) = 10$ and all of its translations $T_1 + t$ are consistent. If A is *False*, only the schedule $T_2(y) = 0$, $T_2(x) = 5$, and $T_2(w) = 15$ and all of its translations are consistent. T_1 and T_2 define the execution strategy St with $St(A) = T_1$ and $St(\neg A) = T_2$. The CTP is obviously not dynamically consistent since we need to know the value of A before execution in order to decide whether we should follow T_1 or T_2, but A is known only after x has been executed in T_1.

However, $H(x, \neg A, T_2) = \neg A$ and so $\neg Con(A, H(x, \neg A, T_2))$. If only this disjunct was used in the definition, the antecedent of the implication would be *False*, and the constraint always satisfied and thus *St* would satisfy the definition, falsely implying the CTP is dynamically consistent.

4. BarN denotes Barber Networks.

References

1. Armando, A., Castellini, C., & Giunchiglia, E. (1999). SAT-based procedures for temporal reasoning. In *5th European Conference on Planning (ECP-99)*.
2. Barber, F. (2000). Reasoning on interval and point-based disjunctive metric constraints in temporal contexts. *Journal of Artificial Intelligence Research*, 12: 35–86.
3. Bessiere, C. (1999). Non-binary constraints. In *Principles and Practice of Constraint Programming (CP'99)*. Springer, Alexandria, Virginia, USA.
4. Chleq, N. (1995). Efficient algorithms for networks of quantitative temporal constraints. In *Proceedings of the Workshop CONSTRAINTS'95*, pages 40–45.
5. Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, 49: 61–95.
6. Georgakopoulos, D., Hornick, M., & Sheth, A. (1995). An overview of workflow management: from process modeling to workflow autonomation infastructure. *Distributed and Parallel Databases*, 3: 119–153.
7. Goldman, R. P., & Boddy, M. S. (1996). Expressive planning and explicit knowledge. In *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems*, pages 110–117.
8. Laborie, P., & Ghallab, M. (1995). Planning with sharable constraints. In *Proceedings of the 14th International Joint Conference on A.I. (IJCAI-95)*, pages 1643–1649.
9. Mackworth, A., & Freuder, E. (1985). The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25(1): 65–74.
10. Morris, P., Muscettola, N., & Vidal, T. (2001). Dynamic control of plans with temporal uncertainty. In *Proceedings of the 17th International Joint Conference on A.I. (IJCAI-01)*, pages 494–499.
11. Muscettola, N., Nayak, P. P., Pell, B., & Williams, B. C. (1998). Remote agent: to boldly go where no AI system has gone before. *Artificial Intellience*, 103: 5–47.
12. Myers, K. L. (1997). Procedural reasoning system: user's guide. Technical report, SRI International.
13. Oddi, A., & Cesta, A. (2000). Incremental forward checking for the disjunctive temporal problem. In *European Conference on Artificial Intelligence (ECAI-2002)*.
14. Onder, N., & Pollack, M. E. (1999). Conditional, probabilistic planning: a unifying algorithm and effective search control mechanisms. In *Proceedings of the 16th National Conference on Artificial Intelligence*, pages 577–584.
15. Peot, M., & Smith, D. E. (1992). Conditional nonlinear planning. In *Proceedings of the First International Conference on AI Planning Systems (AIPS-92)*, pages 189–197. College Park, MD.
16. Pollack, M. E., McCarthy, C., Ramakrishnan, S., Tsamardinos, I., Brown, L., Carrion, S., Colbry, D., Orosz, C., & Peintner, B. (2002). Autominder: a planning, monitoring, and reminding assistive agent. In *7th International Conference on Intelligent Autonomous Systems*, pages 265–272.
17. Pryor, L., & Collins, G. (1996). Planning for contingencies: a decision-based approach. *Journal of Artificial Intelligence Research*, 4: 287–339.
18. Schwalb, E., & Dechter, R. (1997). Processing disjunctions in temporal constraint networks. *Artificial Intelligence*, 93: 29–61.
19. Schwalb, E., Kask, K., & Dechter, R. (1994). Temporal reasoning with constraints on fluents and events. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, Vol. 2, pages 1067–1072. AAAI Press/MIT Press, Seattle, Washington, USA.
20. Smith, D., Frank, J., & Jónsson, A. (2000). Bridging the gap between planning and scheduling. *Knowledge Engineering Review*, 15: 61–94.
21. Stergiou, K., & Koubarakis, M. (2000). Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120: 81–117.
22. Tsamardinos, I., (2001). Constrained-Based Temporal Reasoning Algorithms with Applications to Planning. Ph.D. thesis, University of Pittsburgh, PA.

23. Tsamardinos, I., Morris, P., & Muscettola, N. (1998). Fast transformation of temporal plans for efficient execution. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 254–261.
24. Tsamardinos, I., Pollack, M. E., & Ganchev, P. (2001). Flexible dispatch of disjunctive plans. In *6th European Conference in Planning*, pages 417–422.
25. Tsamardinos, I., Pollack, M. E., & Horty, J. F. (2000). Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling*, pages 264–272.
26. Vidal, T., & Fargier, H. (1999). Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence*, 11: 23–45.